Universal Parallel Analog-to-Digital Encoder Module

T. O. Anderson
Communications Systems Research Section

As more and higher frequency DSIF receiver operations become digitized, the DSN requires higher speed, simpler and more cost-effective analog to digital converters. Besides meeting these requirements the converter described here is modular and lends itself to standardization.

The conversion method used is the all parallel method that is optimum for high speed. One comparator amplifier for each level of the resolution is used. The digital code converter that converts the output of the comparator amplifiers to binary code is included.

The module includes four levels. An eight-level converter is made up of two such modules connected externally, and a sixteen-level converter is made up of four modules externally connected.

I. Introduction

As more and higher frequency DSIF receiver operations become digitized, the DSN requires higher speed, simpler and more cost-effective analog-to-digital (A-D) converters. Besides meeting these requirements the converter described here is modular and lends itself to standardization.

The conversion method used is the all-parallel method, which is optimum for high speed. One comparator amplifier for each level of the resolution is used. The digital code converter from the output of the comparator amplifiers to binary code is included.

The module includes four levels. An eight-level converter is made up of two such modules connected externally and a sixteen-level converter is made up of four modules externally connected.

For reference, five fundamentally different analog-to-digital conversion methods are first reviewed. It will then be clear that the all-parallel conversion method is fundamentally the fastest. The modular code conversion method presented here will make parallel converters more efficient.

The universal converter module presented here includes a resistive ladder, four comparator amplifiers and

four NAND gates. It will quantize an input signal to 2 bits. Two modules, connected externally in a simple manner, will quantize an input signal to 3 bits, four modules to 4 bits, and 8 modules to 5 bits. For higher resolution, a sequential scheme is suggested where universal modules are first connected in parallel, their outputs reconverted to analog form, and the outputs then used as an input to a second set of parallel modules. Most converter requirements can then be satisfied by converters simply implemented as indicated with the universal encoder module being the only component required.

II. A-D Converter Background

As mentioned, five fundamentally different analog-to-digital conversion methods are first reviewed. The parallel method and its features are then discussed. Different methods of code conversion are elaborated on and the optimum modular implementation is shown in detail.

The five fundamentally different types of analog-to-digital converters are:

(1) The chronologic, integration, or counter-ramp type.

(2) The successive approximation type.

(3) The tracking, up/down counter type.

(4) The n-bit, n-series comparator amplifier type.

(5) The n-bit, 2^n comparator amplifier all-parallel type.

Of the first three, each uses one comparator amplifier and one binary weighted D-A ladder network.

In the first type mentioned, the digital-to-analog converter (DAC) is connected to a unidirectional binary counter. The output of the DAC is one input to the comparator amplifier. The second input is the signal to be converted. The output of the comparator amplifier controls the clock gate to the counter. The conversion time is a function of the input signal. For n bits the conversion time is anywhere from zero to $2^n \cdot t_{cl}$ (t_{cl} = clock period). If the start conversion commands are unidistant, the actual sampling instants are not necessarily unidistant. They are however, computable. The sample at the time it is taken, on the other hand, is accurate to the resolution of the converter. This type of converter is inherently slow and cumbersome and not very efficient especially in comparison with more recent designs.

The second type mentioned, the successive-approximation-type converter contains a binary register, a weighted DAC, and a comparator amplifier. The output of the comparator amplifier forms a control bus for the register. The clock steps an n-position sequencer. Starting with the most significant bit in the register, it is turned on, left on, or turned off as a function of the control bus. In some designs a single set of flip-flops is used both as binary register and the sequencer, however with a more complex control logic. Successive approximation control logic designs are described in Ref. 1.

The conversion time for this type of converter is constant, $n \cdot t_{cl}$, however the sample accuracy is a function of the input signal bandwidth. A sample-and-hold amplifier is therefore often required. Nonetheless, this is a moderately fast and moderately efficient converter, and as such it is very popular.

The third type mentioned, the tracking converter, also contains a binary counter, a weighted DAC, and a comparator amplifier. This counter counts in either direction, up or down. Its clock is continuously on and the output from the comparator amplifier controls the counting direction for the counter.

This connection then forms a servo loop that tracks the input signal (as long as the loop slew rate is not exceeded). For a dc input, the output will "dither" between two adjacent states. It will track input changes of 1 least significant bit (LSB) at the rate of the service clock. Digital data can be extracted from the counter/register at any time, preferably in sync with the high-speed service clock that drives the counter. The tracking converter is usually fast, inexpensive, and very useful for the purpose for which it was designed.

The fourth type mentioned, the sequential-comparator-type converter or encoder uses n comparator amplifiers connected in series. The output of the first controls an analog weighting network that subtracts half the input signal from the summing node to the subsequent comparator. The output from the comparator is then directly the output in a binary code. The conversion time is $(n)^{1/2} \cdot t_s$; where t_s is the settling time for one comparator. To maintain a uniform scale factor between stages, the residue can be doubled between stages. This is the fastest of the four basic types of converters mentioned so far and moderately expensive with one comparator per stage.

The last type mentioned, the all-parallel converter or encoder shown in Fig. 1 divides a reference voltage in a resistive divider into 2^n unidistant levels. There are then 2^n comparator amplifiers each with one input from the divider and the other connected to the input signal, the

same as for all the comparators. As the input increases, an increasing number of amplifiers turn on. For a half-scale signal, half of all the amplifiers are turned on (111 . . . 1000 . . . 0). This code is not usable as is, but must be converted to binary in a digital code converter.

The conversion time for this analog-to-digital converter is equal to that of the settling time for a comparator amplifier plus the propagation time through the code converter. It requires 2^n amplifiers, which becomes impractical for large n. For large n, a combination of the all-parallel type and the sequential type is then used. A single-parallel unit for smaller n can also be multiplexed between several analog inputs and digital output registers.

III. Possible Code Converters for the Parallel Encoder

Inspection of the truth tables in Fig. 2 may trigger the thought of connecting exclusive OR gates between A and B, B and C, C and D, etc., and thus first converting the shift code to a unary code. This code can then more easily be converted to binary with a unary-to-binary converter. Exclusive OR gates are available in large-scale integrated (LSI) chips and so is the unary-to-binary converter often called priority encoder. This scheme is not especially efficient. The chip count may be low, but the number of propagation levels high.

One may also discover however, to little avail, that the midway bit of the input code is equivalent to the most-significant bit (MSB) of the output code and that the LSB of the output code is the parity function of the input code, parity coders being readily available in medium-scale integrated (MSI) chips.

One may also discover to little avail that the entire output code is the weight function of the input code. Weighting networks have been explored in Ref. 2. They have usually many propagation levels. They are also too general for this application. They are designed so as not to matter where the ones are located, while in the present case this is known.

IV. Examples of Efficient Code Converters

Straightforward switching networks can be derived as follows: simply by inspecting the code conversion truth tables in Fig. 2, or, by any number of formal methods, one can devise a code converter such as shown in Fig. 3. It may be noticed that in this network, collector OR or wired OR gating is used. This will minimize the number of propagation levels. The implementation then requires open collector gates and common external collector resistors. It may also be noticed that the true output of the converter is the low level.

In Fig. 4 it is shown that the midway bit of the input code is equivalent to the most significant bit of the output code and how this bit partitions both truth tables into two equivalent parts. One can then devise and OR-wire two identical halves and access them with the midway bit. Such a converter is shown in Fig. 5. The access term is in effect an inhibit term for the lower half. Continuing this trend of thought one can of course again partition each half into two identical halves. This partition of the truth table is shown in Fig. 6, and a converter resulting from this strategy is shown in Fig. 7.

A number of other configurations are possible. One could, for example, multiplex one one-half network between two sets of inputs. And of course, one could multiplex one one-quarter network between four sets of inputs. Both of these configurations would take more gates with several more propagation levels.

V. The Universal Code Converter for the Parallel Analog-to-Digital Encoder

The quarter-truth table-converter, if it were to be made up in a special module, can be further streamlined to form a universal module such as shown in Fig. 8. In an iterative network this module can be used to form a code converter for any number of variables.

VI. The Universal Analog-to-Digital Encoder Module

Quad comparator amplifiers in a single module are presently available. The addition of the resistive ladder and the four-NAND-gate code converter will form a universal analog-to-digital encoder module such as shown in Fig. 9. In Fig. 10, four such modules are connected together to form a 4-bit A-D converter.

A 5-bit converter would use eight modules, etc. For greater resolution, a series of parallel converters would be used as discussed. Reconversion to analog form between sets of parallel converters for analog subtraction is then first performed.

VII. Conclusion

Five fundamentally different analog-to-digital conversion or encoding schemes have been reviewed. From this review it is clear that the all-parallel conversion method is the simplest and most efficient. The great variety of available comparator amplifiers allows for designs of encoders of a variety of speeds, resolutions and costs. The most important functional component besides the chain of comparator amplifiers is the code converter. Various designs of the code converter were explored. The optimum modular implementation was discovered.

It has then been shown how a resistive ladder network, four comparator amplifiers, and the optimum code converter module combine to form an optimum universal analog-to-digital encoder modules, which, in turn, combine to form analog-to-digital converters to meet the DSIF requirement stated in the introduction.

References

1. Anderson, T. O., "Optimum Control Logic for Successive Approximation Analog to Digital Converters," *Computer Design*, July 1972, pp. 81–86.
2. Anderson, T. O., "Modular Switching Network for Generating the Weight in Binary Notation of a Binary Vector," *Computer Design*, Apr. 1972, pp. 106–110.

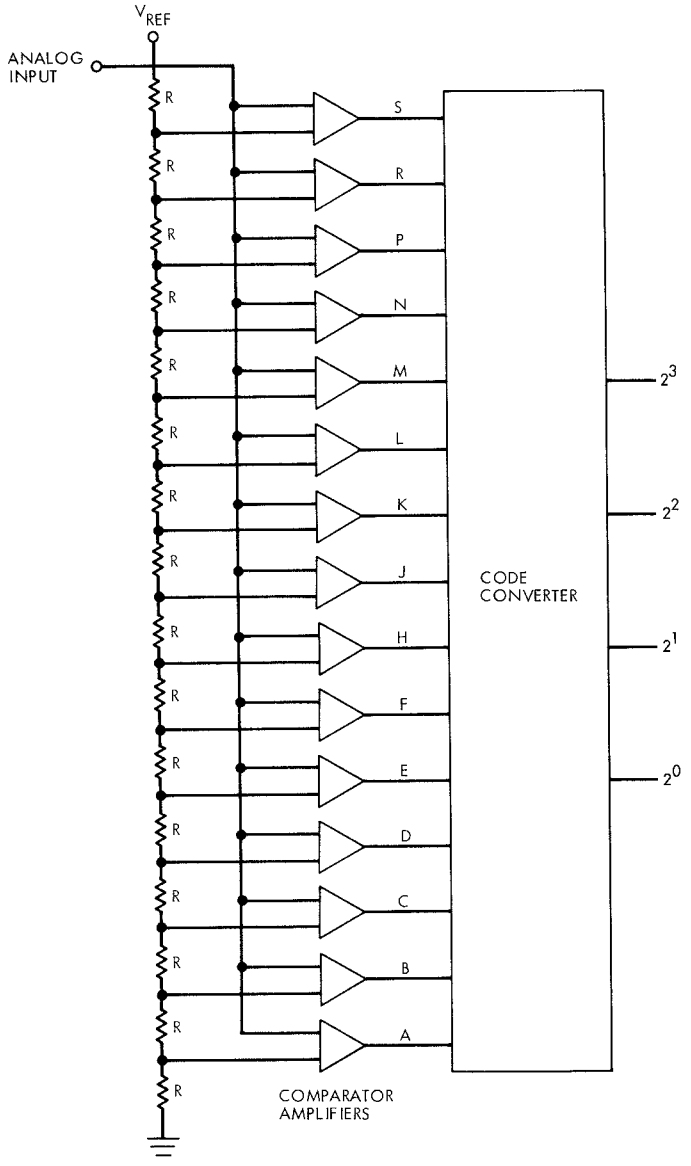

Fig. 1. All parallel code converter block diagram

A	B	C	D	E	F	H	J	K	L	M	N	P	R	S	2^3	2^2	2^1	2^0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	5
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	6
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	7
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	8
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	1	9
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	1	0	10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	1	1	11
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0	12
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	13
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	14
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	15

Fig. 2. Input/output code truth tables

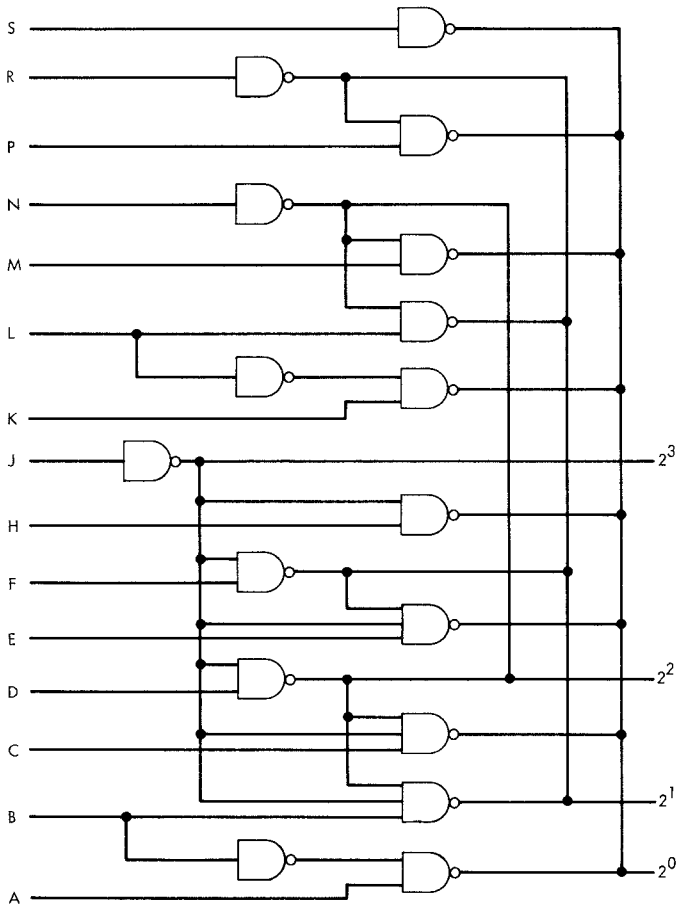

Fig. 3. Code converter logic

A	B	C	D	E	F	H	J	K	L	M	N	P	R	S	2^3	2^2	2^1	2^0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	5
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	6
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	7
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	8
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	1	9
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	1	0	10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	1	1	11
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0	12
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	13
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	14
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	15

Fig. 4. Truth table partitioning

A	B	C	D	E	F	H	J	K	L	M	N	P	R	S	2^3	2^2	2^1	2^0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	3
1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	4
1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1	5
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	0	6
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	1	1	1	7
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	8
1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	0	1	9
1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	1	0	1	0	10
1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1	0	1	1	11
1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	0	0	12
1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	0	1	13
1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0	14
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	15

Fig. 6. Truth table quarter partition

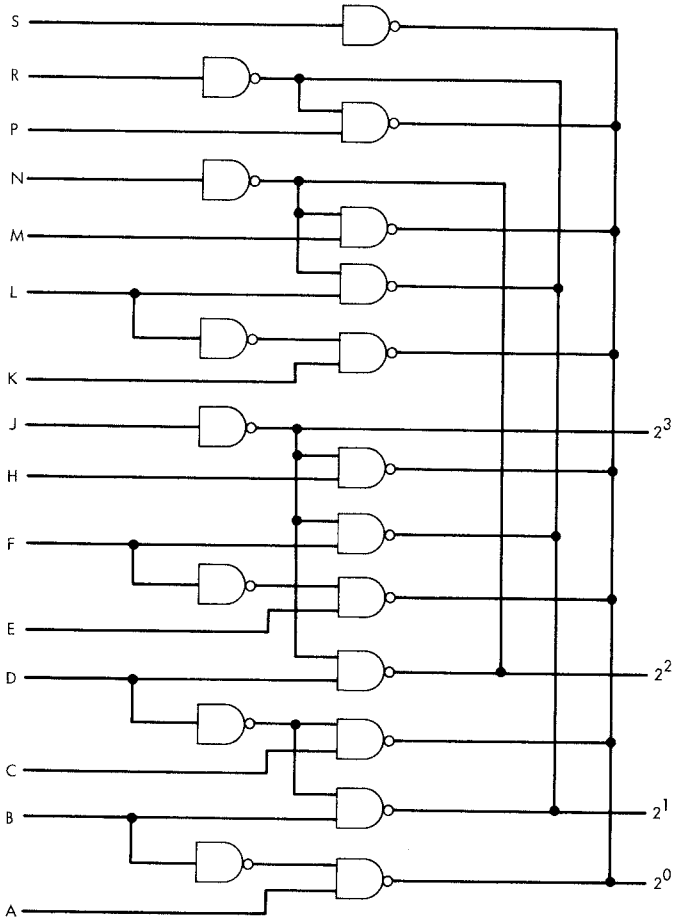

Fig. 5. Two-halves code converter

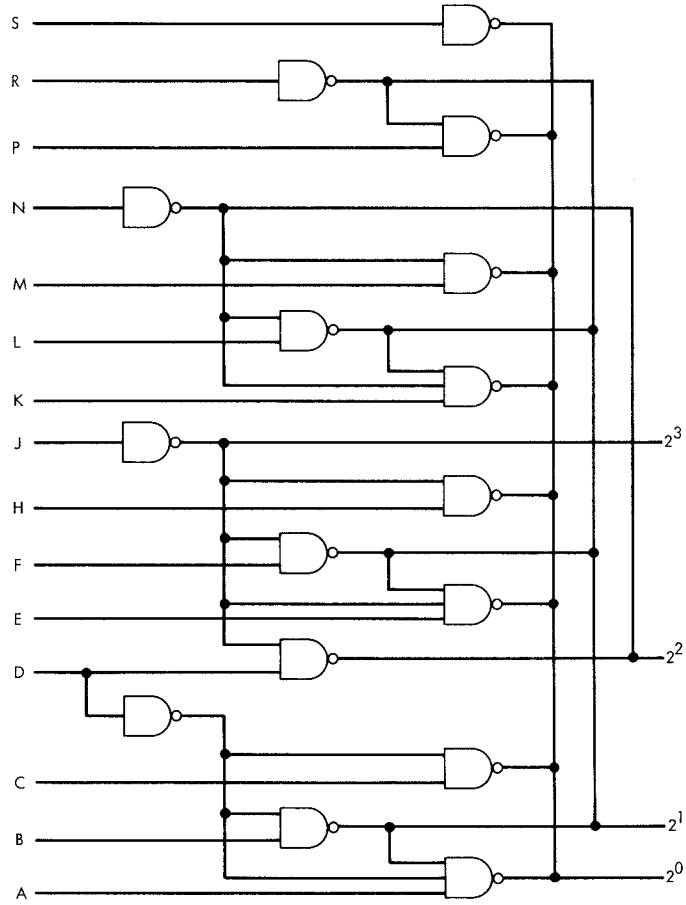

Fig. 7. Four-quarter code converter

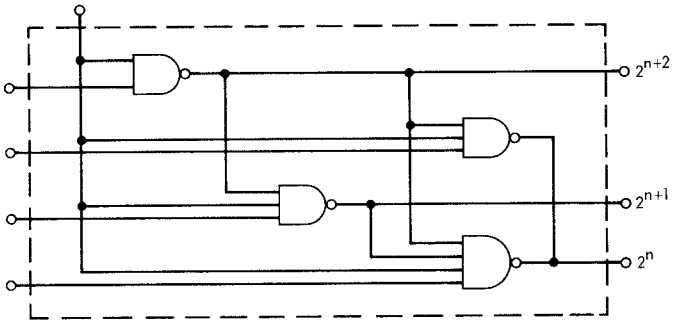

Fig. 8. Universal code converter module

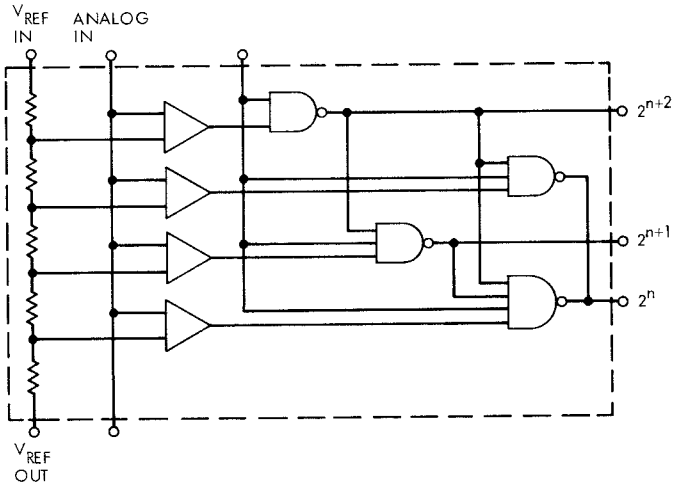

Fig. 9. Universal analog/digital encoder module

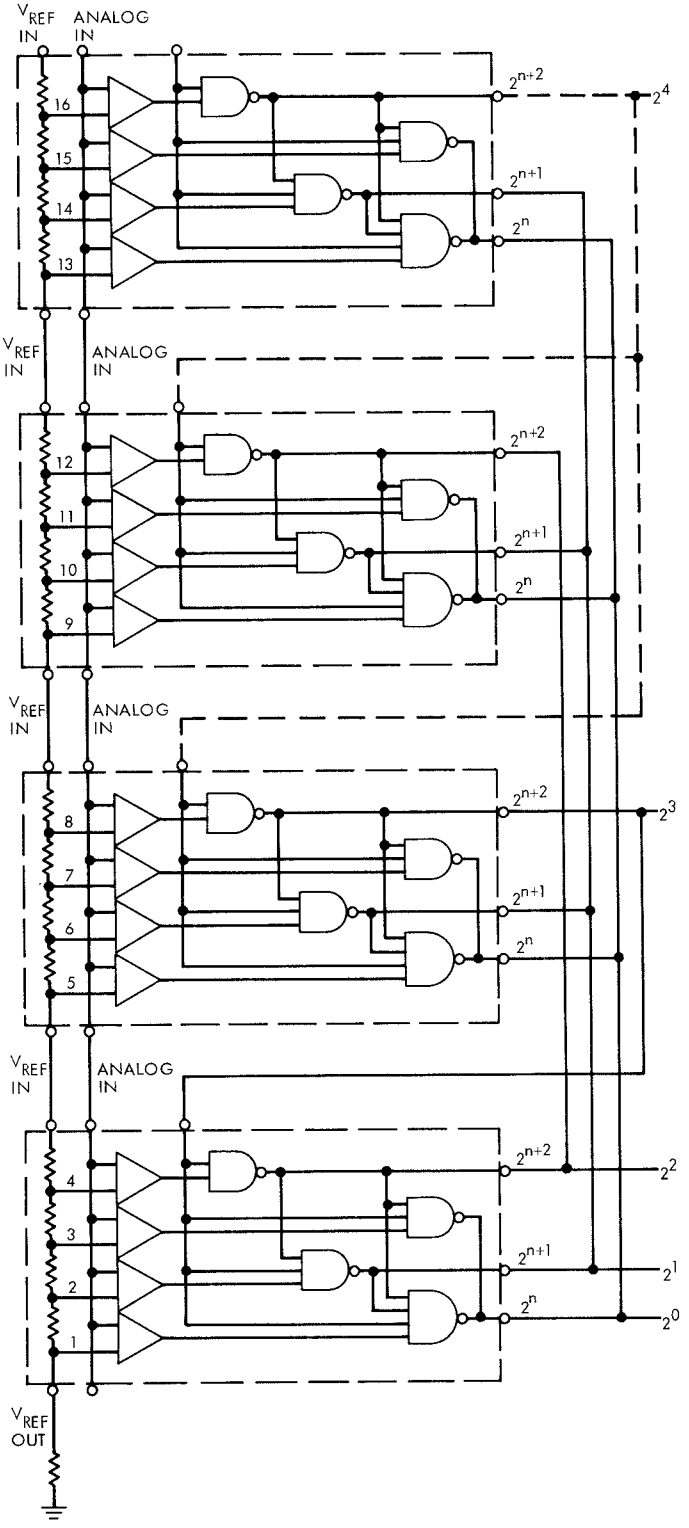

Fig. 10. 4-bit all-parallel A-to-D converter made up of four universal encoder modules